Programming Languages Practice Questions with Answers

Question

C++ code:

```
int x = 10;
x = 3.14;
```

When is the type of x determined? Can the type change later? What happens in this program?

Answer

The type of x is determined at compile time. The type cannot change later. The value 3.14 is converted to an integer, typically 3.

Question

Python code:

```
x = 10
x = 3.14
```

When is the type associated with x determined? Does the type of x change? Is Python statically or dynamically typed?

Answer

The type is associated at runtime. The name x is first bound to an integer object and later to a float object. Python is dynamically typed.

Question

C++ code:

```
int* f() {
  int x = 10;
  return &x;
}
```

What is the lifetime of x? Why is this code unsafe?

Answer

The lifetime of x is limited to the execution of function f. The code is unsafe because it returns the address of a local variable that no longer exists after the function returns. This creates a dangling pointer.

Question

Python code:

```
def f():
  x = 10
  return x
```

When is storage for x created? When is it reclaimed?

Answer

Storage for x is created when the function is called. It is reclaimed after the function finishes and there are no remaining references to the value.

Question

Python code:

```
x = 10

def f():
  return x

def g():
  x = 20
  return f()

print(g())
```

What is the output? Why?

Answer

The output is 10. Python uses static scope. The function f refers to the global x defined outside g.

Question

If the same program were evaluated using dynamic scope, what would the output be? Why?

Answer

The output would be 20. Under dynamic scope, f would use the most recent binding of x in the calling environment.

Question

C++ code:

```
class A {
public:
   void show() { cout << "A"; }
};

class B : public A {
public:
   void show() { cout << "B"; }
};

int main() {
   A* obj = new B();
   obj->show();
}
```

What is printed? Why?

Answer

It prints A. Because the function is not declared virtual, binding happens at compile time.

Question

How would you modify the previous C++ example so that it prints B?

Answer

Declare the function as virtual in class A. This enables dynamic binding at runtime.

Question

Python code:

```
class A:
   def show(self):
      print("A")

class B(A):
   def show(self):
      print("B")
```

```
obj = B()
obj.show()
```

What is printed? When is the method chosen?

Answer

It prints B. Python uses dynamic binding. The method is chosen at runtime based on the object.

Question

C++ block:

```
{
  int x = 5;
}
```

What is the scope of x? What is its lifetime?

Answer

The scope of x is inside the block. Its lifetime is during the execution of that block.

Question

Python code:

```
def f():
   x = 5
   return lambda: x
```

What is the scope of x? Does x continue to exist after f returns?

Answer

The scope of x is inside function f. Yes, x continues to exist because it is captured by the lambda function.

Question

Python code:

```
def make_adder(n):
   def add(x):
      return x + n
```

```
    return add

f = make_adder(10)
print(f(5))
```

What is printed? Why does n still exist?

Answer

It prints 15. The inner function forms a closure and captures the variable n from its defining environment.

Question

C++ declaration:

```
int f(int x);
```

When is the name f bound?

Answer

The name f is bound at compile time.

Question

Python code:

```
def f(x):
    return x + 1
```

When is f created?

Answer

The function f is created at runtime when the interpreter executes the def statement.

Question

Python code:

```
if condition:
    def f(): return 1
else:
    def f(): return 2
```

What happens?

Answer

The function definition is executed at runtime. The name f is bound to one of the two functions depending on the condition.

Question

Define binding.

Answer

Binding is the association between a name and an entity such as a value, type, or storage location.

Question

Define static binding.

Answer

Static binding occurs before runtime, usually at compile time.

Question

Define dynamic binding.

Answer

Dynamic binding occurs at runtime.

Question

Define static scope.

Answer

Static scope means that variable references are resolved using the program's textual structure.

Question

Define dynamic scope.

Answer

Dynamic scope means that variable references are resolved using the calling sequence at runtime.

Question

What is the difference between scope and lifetime?

Answer

Scope refers to where a name is visible in the program text. Lifetime refers to how long the associated storage exists during execution.

Question

Why is static scope easier to reason about?

Answer

Because the bindings can be determined by reading the program text without considering the execution order.

Question

Why can dynamic scope make debugging harder?

Answer

Because variable references depend on the runtime calling sequence, which may not be obvious from the program structure.

Question

What is a closure?

Answer

A closure is a function together with the environment in which it was defined.

Question

Why do closures require static scope?

Answer

Because the function must remember the environment where it was defined, not where it is called.

No answers for these set of questions.

1. Why Study Programming Languages?

Be able to explain:

- Why is studying PL concepts useful beyond learning syntax?
- How does it help in:
  - Choosing languages?
  - Learning new languages?
  - Designing languages?
  - Understanding implementations?

Practice Questions

1. Why is "procedural abstraction" fundamental to programming languages?
2. Give three reasons Sebesta argues PL study improves software development.
3. How does understanding language design improve debugging skills?
4. Why is it dangerous to only know one programming language?

Practice Questions

5. Why did FORTRAN dominate scientific computing?
6. Why is C traditionally used for systems programming?
7. What characteristics make a language good for AI?
8. Why are scripting languages important today?

3. Language Evaluation Criteria

Know and explain:

- Readability
- Writability
- Reliability
- Cost

Practice Questions

9. How does orthogonality improve readability?
10. What language features improve reliability?
11. How does strong typing affect reliability?
12. Explain the trade-off between readability and writability.

4. Influences on Language Design

Be ready to discuss:

- Computer architecture (von Neumann model)
- Programming paradigms
- Software engineering principles

13. What is the von Neumann bottleneck?
14. How does architecture influence imperative languages?
15. Why are paradigms not clean categories anymore?

PART II — PLAI Preface (Critique of Classical View)

(PLAI v3.2.5 Preface, pp. 7–11)

plai-v325

Understand:

- Criticism of "paradigm boxes"
- Programming languages as aggregates of features
- SMoL (Standard Model of Languages)
- SImPl (Standard Implementation Plan)

Practice Questions

16. What does Krishnamurthi criticize about traditional "paradigm" teaching?
17. What is SMoL?
18. Why does PLAI emphasize features instead of paradigms?
19. What is SImPl?
20. Why are "interpreted language" and "compiled language" considered nonsense terms?

PART III — Sebesta Chapter 5 (Names, Bindings, Scope)

This is HIGHLY IMPORTANT for the exam.

1. Names and Variables

Understand:

- Name vs Variable
- Attributes (type, scope, lifetime, value, address)

21. What are the six attributes of a variable?
22. What is a binding?
23. What is static binding?
24. What is dynamic binding?

2. Scope

Know the difference between:

- Static (lexical) scope
- Dynamic scope

25. What is static scope?
26. How does dynamic scope work?
27. Which modern languages use static scope?
28. Why is static scope more predictable?

3. Scope vs Lifetime

29. What is the difference between scope and lifetime?
30. Give an example where lifetime exceeds scope.
31. What are dangling references?
32. What is garbage collection?

4. Referencing Environments

33. What is a referencing environment?
34. How is it determined in static scope?
35. How is it determined in dynamic scope?

5. Named Constants

36. What is a named constant?
37. How does it improve reliability?
38. Difference between manifest constants and dynamic constants?

PART IV — SMoL Tutor (def1, def2, def3, post1)

These focus on:

- Substitution
- Evaluation order
- Static scope
- Environments

Here, I will choose some examples of the tutorial and ask you for the possible output or to trace the stack, the current environment, the scope, etc. (using the stacker format.)